LLaMA

0. Abstract

- LLaMA 모델은 수 조 개의 토큰으로 학습되었으며, 공개적으로 이용 가능한 데이터셋만을 사용하여 모델 훈련 -> 비공개 및 접근 제한 데이터셋을 사용하지 않음
- LLaMA-13B 모델은 GPT-3보다 대부분의 벤치마크에서 더 우수한 성능을 보임

1. Introduction

- LLM은 instructions나 few-shot learning을 기반으로 새로운 작업을 수행할 수 있는 능력 보임
- few-shot learning 능력은 모델을 큰 규모로 확장했을 때 처음으로 나타남
- 매개변수 수가 많을수록 성능이 향상된다고 가정
- 제한된 compute budget에서 최상의 성능을 내는 것은 큰 모델이 아니라 더 많은 데이터로 학습된 작은 모델
- **scailing laws**의 목적은 연산 자원 내에서 데이터셋 크기와 모델 크기를 최적으로 조정하는 방법을 찾는 것
  -> but, 이 접근법은 추론(inference) 비용을 고려하지 않음
- 대규모 언어 모델을 배포할 때는 학습 속도보다 추론 속도가 더 중요함
  -더 작은 모델을 오랜 기간 학습하는 것이 최종적으로 추론 비용을 절감하는 방법이 될 수 있음
- 연구 목표인 "다양한 추론 예산에 맞춰 최상의 성능을 낼 수 있는 모델 개발"을 위해 일반적으로 사용되는 것보다 훨씬 많은 토큰으로 학습 진행

2. Approach

대규모 Transformer 모델을 방대한 양의 텍스트 데이터를 사용하여 학습

2.1 Pre-training Data

- 다양한 도메인의 소스들을 합쳐 학습 데이터셋으로 사용
- 기존 LLMs를 훈련시키는 데 활용한 데이터 소스 재사용, 공개된 데이터만 사용
  English CommonCrawl : 2017년부터 2020년까지 수집된 5개의 CommonCrawl 데이터 덤프를 CCNet 파이프라인(Wenzek et al., 2020)을 이용해 전처리
  -> 중복 제거, fastText 선형 분류기 사용한 non-English 페이지 제거, n-gram 언어 모델을 이용한 저품질 콘텐츠 필터링, Wikipedia에서 참조가 아닌 페이지 삭제
  C4 : 중복 제거와 언어 식별을 포함하여 전처리, CCNet과의 차이점은 품질 필터링(구두점 포함 여부, 단어 및 문장 수와 같은 휴리스틱 방식)
  Github : 휴리스틱 기반으로 라인 길이, 알파벳 숫자 비율 등을 기준으로 저품질 파일 필터링, 정규표현식을 이용하여 header 같은 boilerplate 제거, 파일 단위 중복 제거
  Wikipedia : Latin Cyrillic 문자 체계를 사용하는 20개 언어를 포함한 위키피디아 덤프 추가, 하이퍼링크, 주석, 기타 형식 태그 제거
  Gutenberg and Books3 : 2개의 책 말뭉치 데이터셋 포함, 책 단위 중복 제거 수행, 90% 이상의 동일한 내용을 가진 책 제거
  ArXiv : 과학적인 데이터를 추가하기 위해 arXiv Latex 파일 추가, 논문의 첫 번째 섹션 이전 내용 및 참고문헌 삭제, .tex 파일 내 주석 제거, 사용자가 정의한 매크로 및 정의 확장하여 일관성 증가
  Stack Exchange : 다양한 도메인에 걸친 고품질 질의응답 데이터, HTML 태그 제거, 점수 높은 순으로 답

변 정렬
Tokenizer : Byte Pair Encoding(BPE) 알고리즘 사용, 모든 숫자는 개별 숫자로 분할, 알려지지 않은 UTF-8 문자는 바이트 단위로 분해

boilerplate? 코드를 작성하기 위해 항상 필요한 부분, 여러 상황에서 변경하지 않고 재사용할 수 있는 코드
BPE? 텍스트 데이터를 가변 길이 바이트 쌍으로 표현하여, 빈도가 높은 문자열을 짧은 바이트 쌍으로 나타내는 방식

2.2 Architecture

Transformer 아키텍쳐를 기반으로 구성하고, 기존 Transformer와 차이점이 존재

1. **Pre-normalization**
   학습의 안정성을 향상하기 위해 각 transformer의 sub-layer의 입력을 RMSNorm 정규화
2. **SwiGLU activation function**
   ReLU 활성화함수를 SwiGLU 활성화 함수로 대체, PaLM의 4d 대신 2/3x4d 차원을 이용
3. **Rotary Embeddings**
   Absolute positional embeddings 대신 Rotary positional embeddings(RoPE) 사용

2.3 Optimizer

AdamW 옵티마이저 사용 (하이퍼파라미터: β1 = 0.9, β2 = 0.95)
Cosine learning rate Schedule 적용 -> 최종 학습률이 최대 학습률의 10%가 되도록 설정
weight decay = 0.1, gradient clipping = 1.0, 2000 warmup step

2.4 Efficient Implementation

모델의 학습 속도를 최적화하기 위해 여러 가지 최적화 기법을 적용

1. **효율적인 인과적 다중 헤드 어텐션(Causal Multi-head Attention) 구현**
   - 메모리 사용량 및 실행 시간 단축을 위해 xformers 라이브러리를 활용
   - 어텐션 가중치(Attention weights)를 저장하지 않음
   - 언어 모델링 특성상 마스킹된 Key/Query 점수를 계산하지 않음
2. **체크포인트를 활용한 역전파 최적화 (Checkpointing for Backward Pass)**
   - 역전파 시 재계산해야 하는 활성값의 양을 줄임.
   - 선형 계층의 출력과 같이 계산 비용이 높은 활성값을 저장하여 재사용
   - PyTorch autograd를 그대로 사용하지 않고 Transformer 레이어의 역전파 함수를 직접 구현
3. **모델 및 시퀀스 병렬 처리(Model & Sequence Parallelism) 적용**
   - All-reduce 연산을 활용해 GPU 네트워크 내에서 활성값 계산과 데이터 전송이 동시에 수행되도록 구성

3. Main results

Zero-shot 학습

- 모델 주어진 작업의 텍스트 설명과 테스트 예제를 제공
- open-ended generation 방식으로 답을 생성하거나, 제공된 답변 옵션을 랭킹
  Few-shot 학습
- 1개에서 64개 사이의 예제와 테스트 예제를 제공

- 모델이 이 이 입력을 기반으로 답을 생성하거나, 여러 선택지 중에서 가장 적절한 답을 랭킹

LLaMA 모델을 free-form generation 태스크 및 multiple choice 태스크에서 평가
Multiple choice tasks(객관식 문제 평가 방식)

- 주어진 문맥을 바탕으로 가장 적절한 답을 선택해야 함
- 문맥과 답변 간의 확률이 가장 높은 것을 선택
- P(completion|context) / P(completion|“Answer:”) 방식으로 정규화하여 답을 선택

4. Instruction Finetuning

- instruction data로 간단히 파인튜닝만 해도 MMLU 성능이 빠르게 향상됨
- 파인튜닝 이후 모델의 명령어 수행 능력이 더욱 향상됨

5. Bias, Toxicity and Misnformation

- LLMs는 학습 데이터에 존재하는 편향을 재현하고 증폭할 수 있음
- 유해한 콘텐츠를 생성할 가능성 존재
- 모델의 크기가 클수록 유해성 점수가 증가하는 경향이 관찰됨
- 특히, Respecful prompt에서 유해성이 증가하는 현상

7. Related Work

- **Language Model**은 단어, 토큰 문자 시퀀스의 확률 분포를 학습하는 모델
- 다음 토큰의 예측이 NLP의 핵심 과제로 연구됨

8. Conclusion

- 최신 기초 모델(Foundation Models)과 경쟁할 수 있는 대형 언어 모델을 공개
- 비공개 데이터를 사용하지 않고, 공개적으로 이용 가능한 데이터만으로도 SOTA를 달성할 수 있음을 입증